IN THE CLAIMS

Please amend the claims as shown below.

1. (Currently Amended) A digital information product comprising:

a computer-readable medium; and

stored on the computer-readable medium, computer program instructions defining a software system that produces software code based on a set of functional requirements and design parameters provided by a programmer, wherein the computer program instructions, when executed, allow the programmer to define a design matrix describing a relation between the set of functional requirements and the design parameters.

2. (Canceled)

3. (Currently Amended) The digital information product according to claim [[2]] 1, wherein the computer program instructions, when executed, allow a programmer to manipulate the design matrix into lower triangular form.

a2

4. (Currently Amended) The digital information product according to claim [[2]] 1, wherein the computer program instructions, when executed, allow a programmer to determine a decoupled design matrix.

5. (Currently Amended) The digital information product according to claim [[2]] 1, wherein the computer program instructions, when executed, allow a programmer to determine an uncoupled design matrix.

6. (Currently Amended) A method for producing a software system, the method comprising:

defining a design matrix describing a relation between functional requirements of [[the]] a software system and design parameters; and

generating software code based on the design matrix.

7. (Original) The method according to claim 6, further comprising a step of constructing functions using diagonal elements of the design matrix.

8. (Original) The method according to claim 6, further comprising a step of defining a flow description that describes a flow of the software system.

9. (Original) The method according to claim 6, further comprising a step of defining a software module corresponding to at least one functional requirement.

10. (Original) The method according to claim 9, wherein the software module is defined based upon the flow description.

11. (Original) The method according to claim 6, further comprising a step of defining a plurality of software modules, each of which corresponds to a respective functional requirement.

12. (Original) The method according to claim 11, further comprising relating the plurality of software modules by junctions.

13. (Original) The method according to claim 12, further comprising a step of combining the plurality of software modules according to their junctions.

14. (Currently Amended) The method according to claim 13, wherein at least one of the junctions is selected from a group including:

a summation junctions;

a control junction; and

a feedback junction.

15. (Currently Amended) A method for producing software comprising:

defining a design matrix describing a relation between functional requirements of [[the]] a software system and design parameters so that the design matrix has a lower triangular form; and

generating software code based on the design matrix.

16. (Canceled)

17. (Original) The method according to claim 15, wherein the step of defining includes determining an uncoupled design matrix.

18. (Original) The method according to claim 15, wherein the step of defining includes determining a design matrix having a diagonal form.

19. (Original) The method according to claim 15, wherein the step of defining includes determining a decoupled design matrix.

Q1

20. (Original) The method according to claim 15, wherein the step of generating comprises:

generating a plurality of models; and

outputting the models to a code generator.

21. (Original) The method according to claim 15, wherein the step of generating comprises generating at least one diagram describing an object-oriented object structure.

22. (Original) The method according to claim 21, wherein the at least one diagram is in a Unified Modeling Language format.

23. (Original) The method according to claim 21, wherein the at least one diagram includes at least one of a group of diagrams comprising:

a use-case diagram; and

an entity-relation diagram.

24. (Original) The method according to claim 15, wherein the code generated includes object-oriented entities.

25. (Original) The method according to claim 15, further comprising a step of defining an object-oriented class using at least one functional requirement.

26. (Original) The method according to claim 24, further comprising a step of defining a child of the object-oriented class using at least one functional requirement.

27. (Original) The method according to claim 25, further comprising a step of defining a grandchild of the object-oriented class using at least one functional requirement.

28. (Original) The method according to claim 27, wherein the grandchild of the object-oriented class defines behavior of the class.

29. (Original) The method according to claim 24, wherein at least one functional requirement represents at least one object-oriented software entity.

30. (Original) The method according to claim 15, wherein the generated software code is Source Code.

31. (Currently Amended) A method for designing software involving object-oriented software objects, the method comprising:

defining a design matrix describing a relation between a plurality of functional requirements of the software system and design parameters;

representing at least one object-oriented object by at least one of the functional requirements;

representing data used by the at least one object-oriented software object by at least one of the design parameters; and

representing a method of the at least one object-oriented software object by a product of a portion of the design matrix [[an]] and the at least one design parameter.

32. (Original) The method according to 31, further comprising defining a plurality of functional requirements and constraints of the software system.

33. (Original) The method according to 31, comprising a step of decomposing at least one functional requirement into leaf elements.

34. (Original) The method according to 31, comprising a step of decomposing at least one design parameter into leaf elements.

35. (Original) The method according to 31, comprising a step of decomposing at least one process variable into leaf elements.

36. (Original) A method for designing software comprising:

defining a software system by defining a design matrix describing a relation between functional requirements of the software system and design parameters implementing the software system;

defining at least one object-oriented object; and

defining at least one method that may be defined on the at least one object, wherein the at least one object-oriented object and the at least one method are related to the design matrix.

37. (Original) The method according to claim 36, further comprising defining an object class of the at least one object-oriented object that represents a first level of functional requirements of the design matrix.

38. (Original) The method according to claim 37, further comprising defining an instance of the object class represented by a second level of functional requirements of the design matrix.

39. (Original) The method according to claim 38, further comprising defining behavior of the object class as a third level of functional requirements of the design matrix.

40. (Original) The method according to claim 39, wherein the first, second, and third levels are successive levels of a functional requirement hierarchy.

41. (Original) The method according to claim 36, further comprising mapping the functional requirements into a physical implementation domain by defining design parameters.

42. (Original) The method according to claim 36, further comprising a step of construction functions using diagonal elements of the design matrix.

a1

43. (Original) The method according to claim 36, wherein the step of defining a software system includes determining whether the design matrix is uncoupled.

44. (Original) The method according to claim 36, further comprising generating a plurality of models and outputting the models to a code generator.

45. (Original) The method according to claim 44, wherein the step of generating comprises generating at least one diagram describing an object-oriented system structure.

46. (Original) The method according to claim 45, wherein the at least one diagram includes at least one of a group of diagrams including:

a use-case diagram; and

an entity-relation diagram.

47. (Original) The method according to claim 45, wherein the at least one diagram is in a Unified Modeling Language format.

48. (Original) A database format for designing a software system comprising:

a software design specification, wherein the design specification defines a design matrix describing a relation between a plurality of functional requirements of the software system and design parameters, wherein the design specification represents at least one software object by at least one of the functional requirements, and wherein the design specification represents data used by the at least one software object by at least one of the design parameters; and

software code produced by the design specifications.

49. (Original) The database format according to claim 48, wherein the software design specification further comprises a design identification identifying the software code.

50. (Original) The database format according to claim 48, wherein the software design specification further comprises a detailed design description including the design matrix, functional requirements and design parameters.

51. (Original) The database format according to claim 49, wherein the software design specification further comprises a detailed design description including the design matrix, functional requirements and design parameters.

52. (Canceled)

53. (Original) The database format according to claim 48, wherein the design parameters are arranged in a plurality of levels, and wherein each of the plurality of levels can be referenced by another software system.

54. (Original) The database format according to claim 48, wherein the database format for designing a software system is used in a software production system, and a user

operating the software production system is capable of creating software comprising one or more elements of the software system.

55. (Original) The database format according to claim 54, wherein the software production system further comprises a user interface and a database implementing the database format, and the software production system is capable of searching through the database to obtain one or more elements of the software system to create a new software system.

56. (Original) The database format according to claim 55, wherein a new element of the new software system interfaces with one or more elements of the software system, and the software production system is operable to verify that the interface is operable.

57. (Original) The database format according to claim 55, wherein the software production system is configured to verify a consistency of design of the new software system.

58. (Original) The database format according to claim 55, wherein the software production system is configured to search design parameter information, including at least one of description information, keywords associated with the software system, categories in which the software system is assigned, and comments.

59. (Original) The database format according to claim 48, wherein the software code is in executable form.

60. (Original) The database format according to claim 48, wherein the software code is source code.

61. (Original) The database format according to claim 57, wherein the software production system is configured to verify a consistency of design of the new software system by checking a consistency between a first level of design implementing a design parameter from the database and a second level of design of the new software system.

62. (Original) The database format according to claim 61, wherein the second level of design is a child of the first level of design.

63. (Original) The database format according to claim 55, wherein a new element of the new software system interfaces with one or more elements of the software system, and the software production system is operable to verify that the interface is operable.

64. (Currently Amended) A database format for designing a software system comprising:

design identification information identifying information describing the software system;

detailed design description information describing the structure and operating qualities of the software system, wherein the detailed design description information defines a design matrix describing a relation between a plurality of functional requirements of the software system and design parameters, represents at least one software object by at least one of the functional requirements, and represents data used by the at least one software object by at least one of the design parameters; and

software code information associated with the software system.

65. (Canceled)

66. (Original) The database format according to claim 64, wherein the software code information comprises source code.

67. (Original) A method for generating software code associated with a software system, the method comprising:

defining a design matrix describing a relation between functional requirements of the software system and design parameters; and

generating software code based on the design matrix.

68. (Original) The method according to claim 67, further comprising defining a plurality of functional requirements and constraints of the software system.

69. (Original) The method according to claim 67, further comprising a step of decomposing at least one functional requirement into leaf elements.

70. (Original) The method according to claim 67, further comprising a step of decomposing at least one design parameter into leaf elements.

71. (Original) The method according to claim 67, further comprising a step of decomposing at least one process variable into leaf elements.

72. (Original) The method according to claim 67, wherein the software code is written in a modeling code language.

a1

73. (Original) The method according to claim 67, wherein the software code is executable by a computer system.

74. (Original) The method according to claim 72, wherein the modeling code language is the Unified Model Language (UML).

75. (Original) The method according to claim 72, wherein the software code describes at least one diagram of a group comprising:

a class diagram;
an interaction diagram;
a collaboration diagram;
a sequence diagram;
a state diagram; and
an activity diagram.

76. (Original) The method according to claim 75, further comprising a step of relating the design matrix to at least one element of the at least one diagram.

77. (Original) The method according to claim 72, wherein the system creates a use-case diagram based upon the design matrix.

78. (Original) The method according to claim 67, further comprising defining a plurality of customer needs, the plurality of needs being arranged in a hierarchical tree having levels, and wherein at least one of the levels relates to a layer of a use-case diagram.

79. (Original) The method according to claim 67, wherein the use-case diagram includes a relation layer, a use case layer and an actor layer.

80. (Original) The method according to claim 79, wherein the actor layer represents one or more actors.

81. (Original) The method according to claim 79, wherein the use case layer describes customer needs that are attributed to the actor.

82. (Original) The method according to claim 79, wherein the relation layer describes a relationship between customer needs in customer needs in the use case layer.

83. (Original) The method according to claim 82, wherein the relationship is a semantic connection among model elements.

84. (Original) The method according to claim 82, wherein the software code includes a model file which describes the modeling code.

85. (Original) The method according to claim 84, wherein the model file is suitable for use as an input file for use by a software production system to create a software system.

86. (Original) A computer-readable medium encoded with a program that, when executed on a computer system, performs a method for rendering an interface through which a user may interact, the method comprising steps of:

displaying a software design interface, wherein the interface displays a set of functional requirements upon which a software design is based, and wherein the interface displays a design matrix describing a relation between the set of functional requirements and design parameters implementing the software design.

87. (Original) The computer-readable medium according to claim 86, wherein the computer system performs a step of indicating, in association with the displayed design matrix, whether the software design specified by the design matrix is at least one of a group including a decoupled, uncoupled or coupled design.

88. (Original) The computer-readable medium according to claim 86, wherein the design matrix represents a coupled design, and wherein the computer system performs a step of indicating, in association with the displayed design matrix, whether a design parameter associated with the design matrix is coupled.

89. (Original) The computer-readable medium according to claim 88, wherein the computer system performs a step of indicating, for the design parameter associated with the design matrix, an alternate rearrangement of the design parameter wherein the design parameter, if rearranged, is uncoupled or decoupled.

90. (Original) The computer-readable medium according to claim 88, wherein the computer system performs a step of indicating, for the design matrix, an accuracy of the software design with respect to an optimum design.

91. (Original) The computer-readable medium according to claim 86, wherein the computer system performs a step of indicating, for the design matrix, a range of acceptable inputs over which the design matrix operates.

92. (Original) The computer-readable medium according to claim 86, wherein the computer system performs a step of arranging and displaying the functional requirements and design parameters to the user in a hierarchical manner.

93. (Original) The computer-readable medium according to claim 92, wherein the computer system performs a step of displaying the functional requirements and design parameters in a hierarchical tree having a plurality of levels.

94. (Original) The computer-readable medium according to claim 90, wherein the computer system performs a step of indicating a robustness of the software.

95. (Original) The computer-readable medium according to claim 92, wherein the computer system performs a further step of creating, based upon an arrangement of the design parameters, a Gantt chart describing production dependencies of the software design.